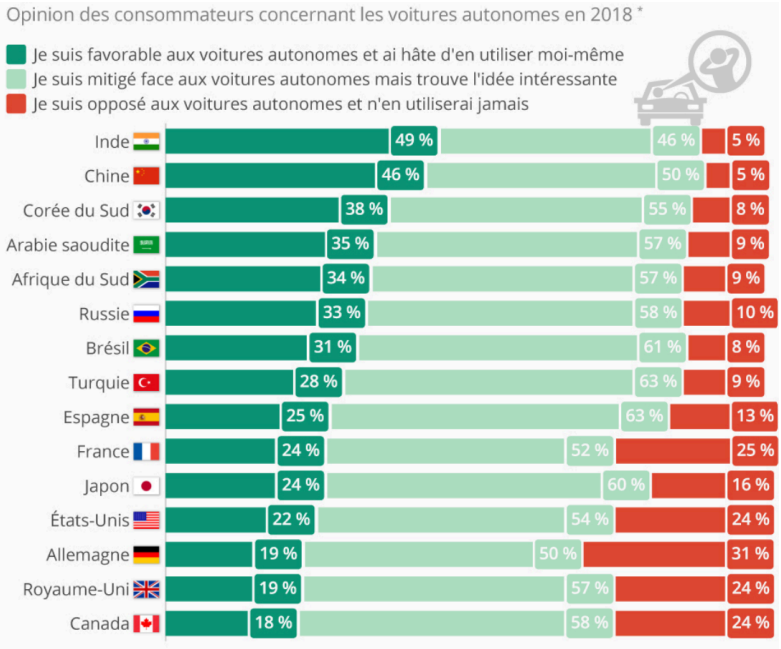
Optimisation Bi-niveau

Détection de voitures dans le trafic

Hoël Roquinarc'h
Paul Hercouët
Come Cambien
Théo Manea
Oleksandr Vladimirov

I - Introduction/Problématique

Avec l'engouement grandissant vis-à-vis des voitures autonomes durant ces dernières années, la marché devient de plus en plus intéressant pour les entreprises du milieu automobile. La détection des environnements, étant un enjeu de sécurité indispensable, gagne une importance encore jamais vue auparavant.



Une interrogation majeure actuelle est donc de savoir s'il est possible, en partant d'une image donnée, de détecter où se trouvent les autres véhicules.
Ce projet a donc pour objectif de détecter, à partir d'images de trafic quelconque, où se trouvent les voitures.

II - Data utilisée

Nous avons travaillé en utilisant deux bases de données lors de ce projet :

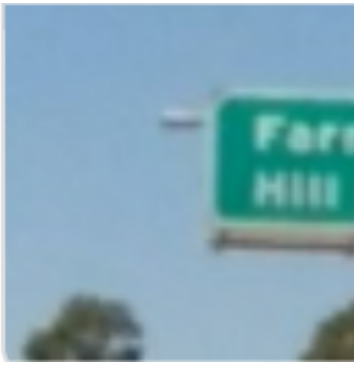
I - Bases de données d'images de voitures

Dataset : https://www.kaggle.com/datasets/brsdincer/vehicle-detection-image-set

Cette base de données contient un ensemble d'image de 64 x 64 pixels séparées en deux catégories :

- des images de voitures sur des routes quelconques

- des images de routes sans voitures avec parfois des panneaux pour dérouter le système



Nous avons utilisé cette base de données durant la première partie du projet afin d'entraîner le modèle de détection de voiture utilisé dans le projet.

II - Bases de données d'image de trafic

Dataset : https://www.kaggle.com/datasets/ashfakyeafi/road-vehicle-images-dataset

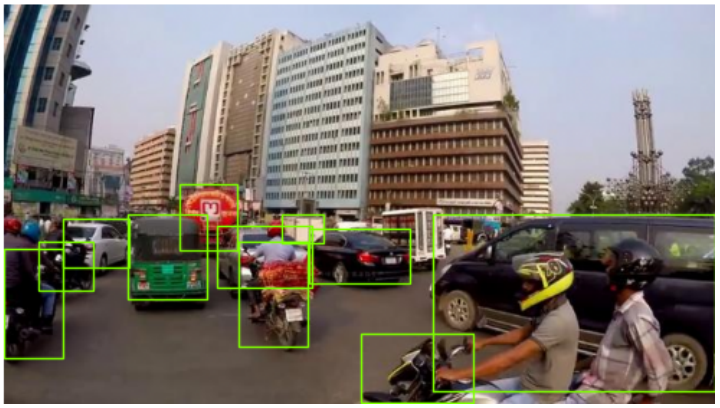
Cette base de données contient plusieurs éléments:

- des images de trafic routier pouvant contenir des véhicules
- des labels associés à celles-ci indiquant la position des véhicules dans l'image en utilisant le format YOLO v5 pour les décrire
  exemple de label YOLO v5 :

```
18 0.7578125 0.36666666666666664 0.259375 0.6555555555555556
class_id x_center y_center width height
```

Contrairement à la base de données précédente, les images n'ont pas une taille fixe.

Nous avons détouré les images en utilisant les coordonnées données par les labels pour voir à quoi ressemblaient les informations des labels.



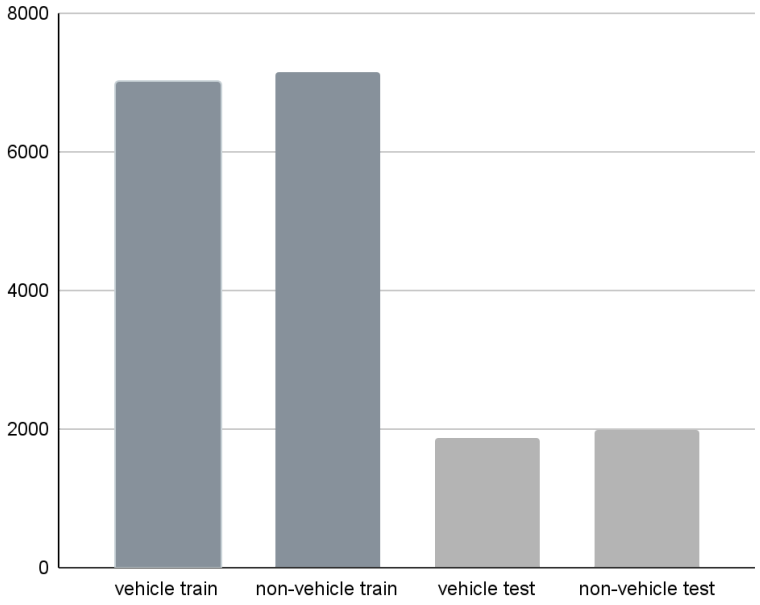
Cette base de données été utilisée dans la deuxième partie de notre projet où l'on utilise le modèle précédemment entraîné afin de détecter où se trouvent les voitures

Ces images avec la localisation des véhicules (voir image ci-dessus) seront utilisées pour comparer avec les images produites par notre modèle.

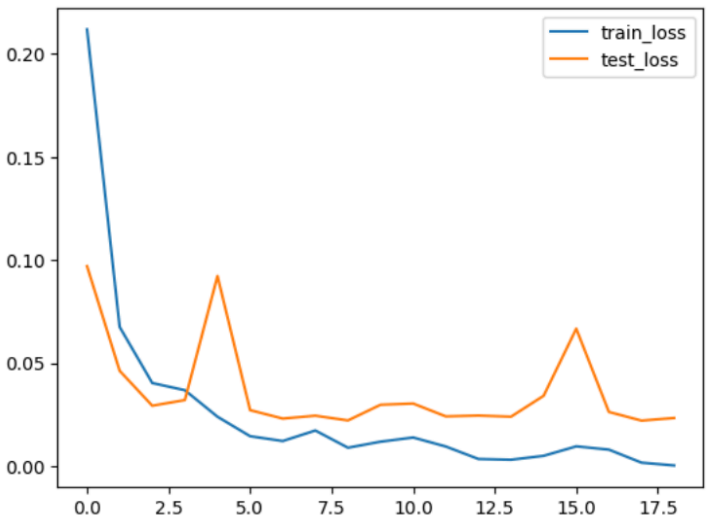
III - Système de détection

Le modèle de détection est en premier lieu entraîné sur le premier dataset. C'est avec lui que le modèle "apprend" à reconnaître les voitures.

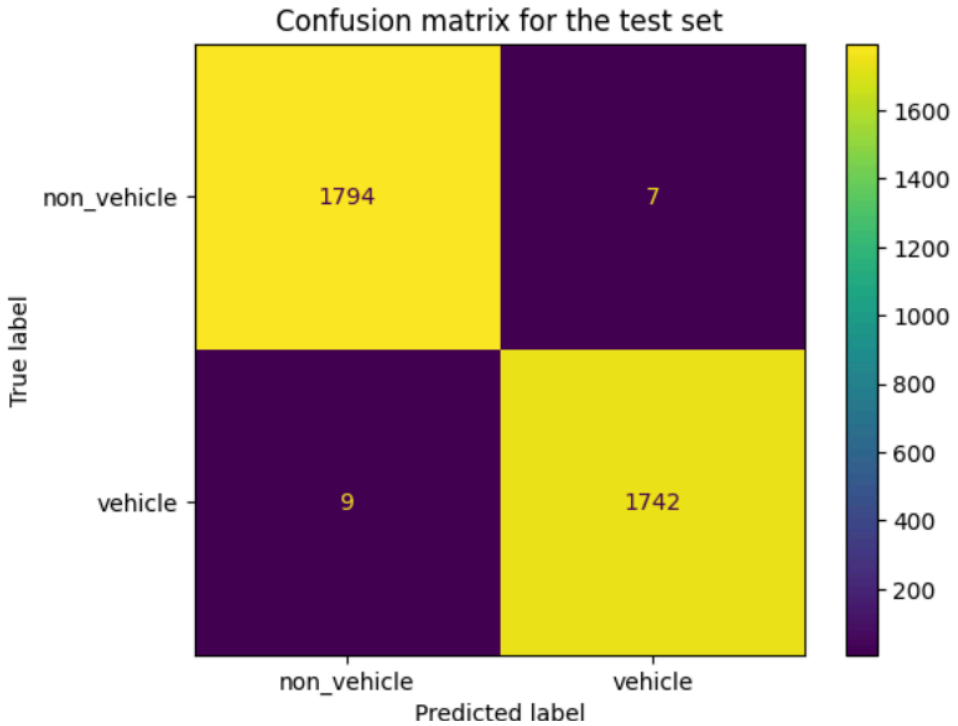
Le dataset est réparti de la façon suivante :



Chacune des parties, véhicules et non véhicules, est séparée en 2 sous-parties, l'une réservée à l'entraînement (les parties train) et l'autre pour pour les tests (les parties test).

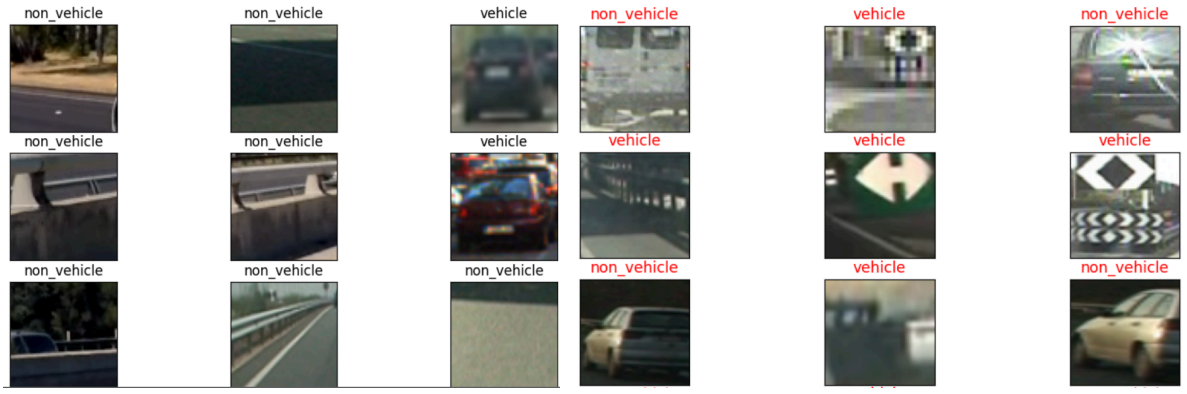


Le modèle est entraîné pendant 20 epochs, l'objectif est de faire tendre la train_loss vers 0. On obtient après l'entraînement, une train_loss de moins de 0.05 et celle-ci commence à se stabiliser doucement, donc il est préférable d'arrêter l'entraînement pour éviter l'overfitting.



On obtient, après l'entraînement, la matrice de confusion ci-dessus. On obtient 7 faux positifs et 9 faux négatifs.

Les images ci-dessous présentent des cas avec le label attribué par le modèle ainsi que l'image correspondante. L'image de gauche correspond à des images qui ont été correctement labellisées et celle de droite à des images qui ont été mal labellisées.



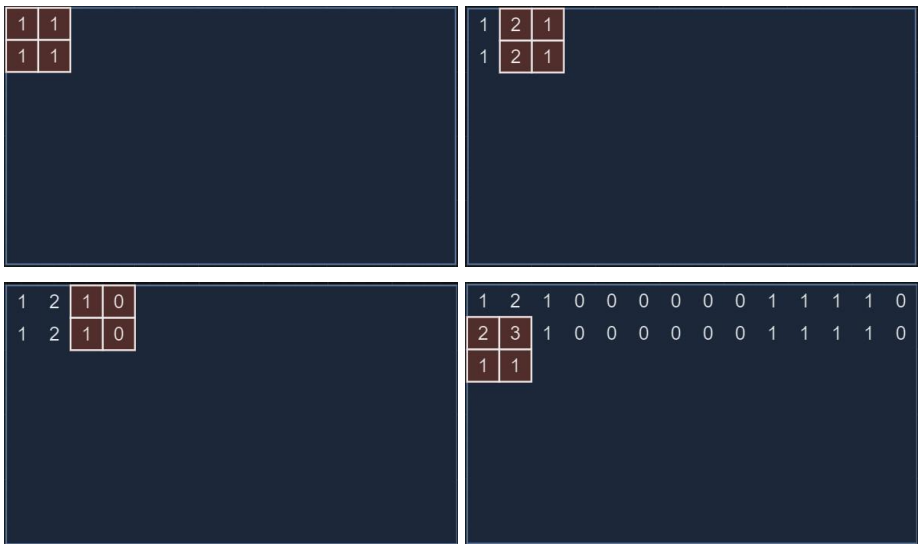
IV - Application sur les images de trafic

Une fois le modèle entraîné, on va l'appliquer sur les images du second dataset, celles qui contiennent des images de véhicules dans le trafic.

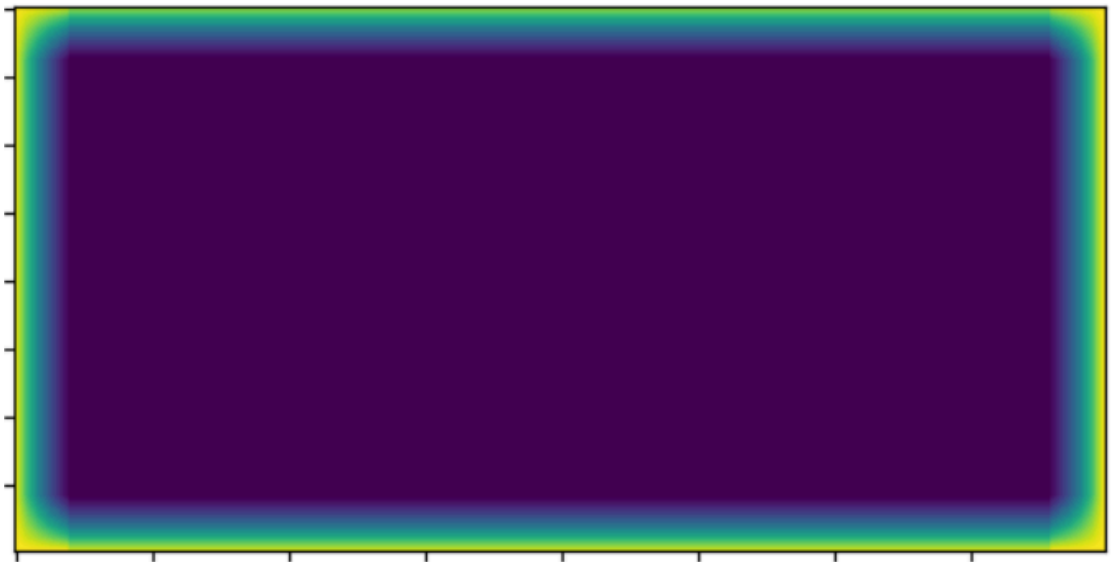
Voici ci-dessous un schéma simple qui explique comment fonctionne l'algorithme:

Nous avons une image résultat qui fait la même taille que notre image à analyser mais est remplie de zéros.

Une fenêtre d'une taille définie va défiler sur l'écran (ici une taille de 2x2). La fenêtre avance sur l'image avec un pas définie (en nombre de pixels).A chaque étape la partie de l'image qui est dans la fenêtre va être analysée par notre réseau de neurones, si le réseau considère qu'il y a une voiture sur l'image alors nous allons ajouter 1 à tous les points de l'image résultat qui sont dans la fenêtre.on fait ensuite coulisser la fenêtre et on recommence.



Cela amène un nouveau problème : les bords de l'image sont vus par moins de fenêtres que le reste des pixels de l'image. Pour cela nous avons créé un masque d'image que nous allons multiplier à notre image finale pour favoriser les bords. Le masque d'image aura des valeurs entre 1 et 0 pour pouvoir égaliser les chances d'être choisi.
Pour créer ce masque nous avons compté le nombre de fenêtre que va voir chaque pixel et par la suite nous avons appliqué la fonction $f(x) = \frac{1}{e^x}$ à chaque élément de ce pré-masque.



V - Optimisations

Dans le cadre de l'optimisation du modèle, nous avons testé différents modèles avec des paramètres différents pour un total de 72 modèles (voir tableau ci-dessous)

	type	w_size	step	parameter	score	false_positive	false_negative
pred_64_8_raw.pkl/quant0.6	quant	64	8	0.6	0.639458	0.280752	0.111237
pred_128_32_raw.pkl/quant0.6	quant	128	32	0.6	0.585376	0.263764	0.118915
pred_64_8_raw.pkl/quant0.7	quant	64	8	0.7	0.539845	0.207262	0.132288
pred_128_32_raw.pkl/val0.3	val	128	32	0.3	0.502595	0.220123	0.125205
pred_64_32_raw.pkl/quant0.6	quant	64	32	0.6	0.475559	0.188714	0.137261
pred_128_32_raw.pkl/quant0.7	quant	128	32	0.7	0.473977	0.180848	0.139732
pred_128_64_raw.pkl/quant0.6	quant	128	64	0.6	0.431864	0.205449	0.141754
pred_64_8_raw.pkl/val0.3	val	64	8	0.3	0.422677	0.146041	0.142849
pred_128_64_raw.pkl/val0.3	val	128	64	0.3	0.414333	0.194913	0.141228
pred_64_8_raw.pkl/quant0.8	quant	64	8	0.8	0.411628	0.131322	0.155278

Les paramètres qui varient sont :

- le type (quantile ou valeur)
- la taille de la fenêtre prise pour scanner l'image (w_size)
- le pas entre chaque fenêtre (step)
- le paramètre propre à quantile et valeur (parameter)

Pour chacun des modèles, on compare les résultats du modèle avec les images extraites du second dataset pour lesquelles, nous avons la position des véhicules.

Le colonne score correspond à la formule suivante :

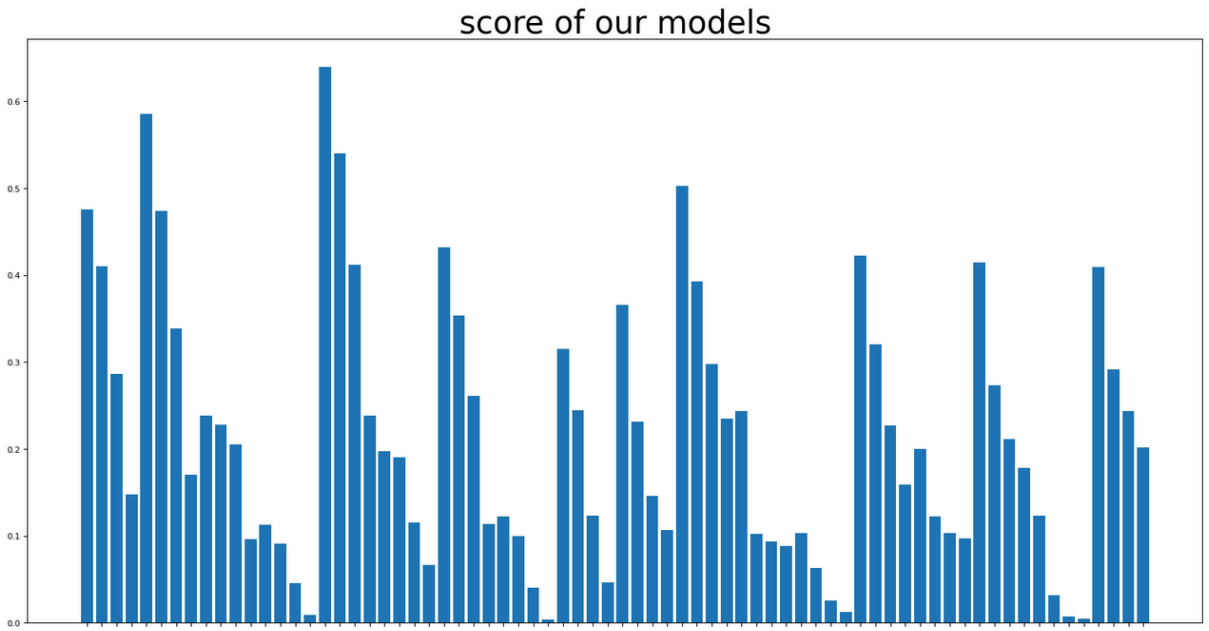
$$\frac{Nb\ pixels\ commun}{Nb\ pixels\ YOLO}$$

Le résultat de cette formule est un pourcentage

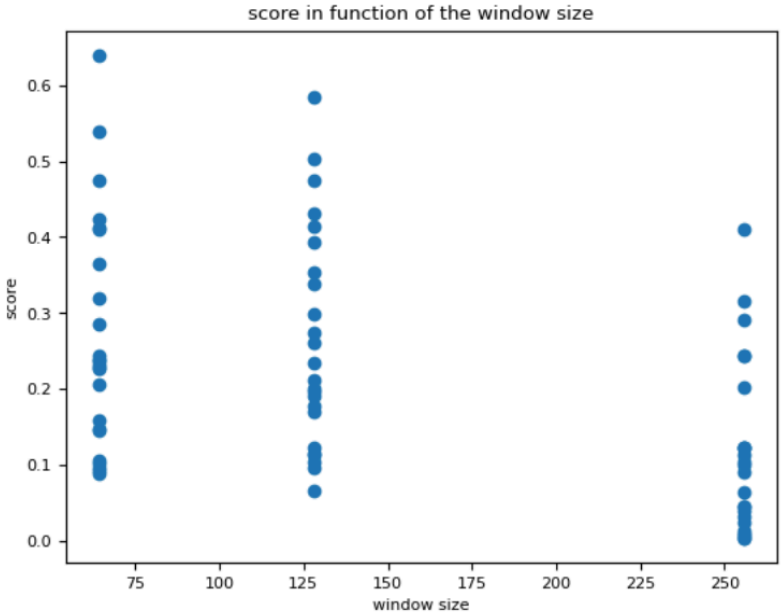
$Nb\ pixels\ commun$ correspond au nombre de pixels sélectionnés par notre modèle comme appartenant à un véhicule qui sont bel et bien considérés comme faisant partie d'un véhicule d'après les données extraites du second dataset.

$Nb\ pixels\ YOLO$ correspond au nombre total de pixels faisant partie de véhicules d'après les données sous le format YOLO du second dataset.

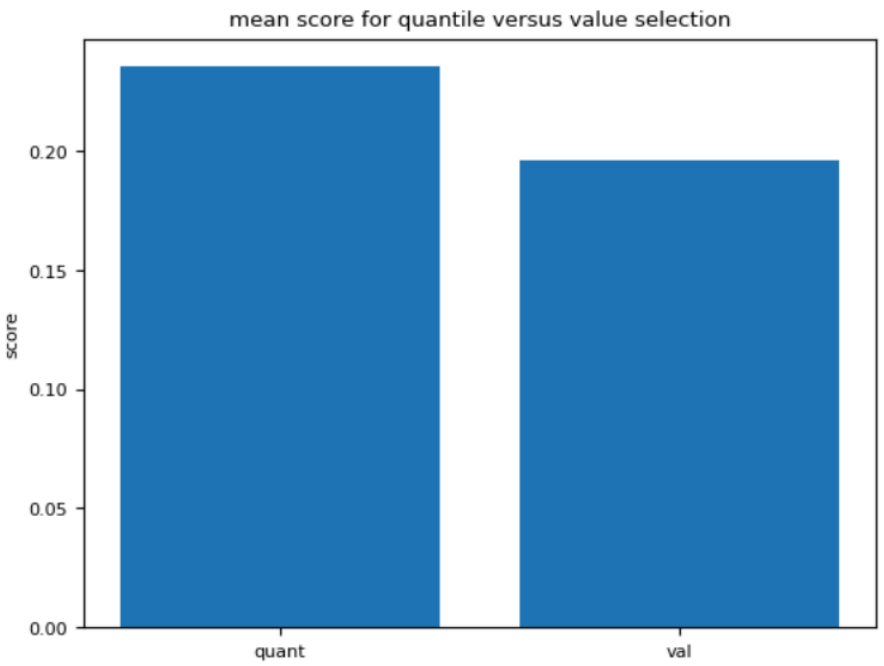
A partir de ce score, on peut déterminer les modèles plus efficaces parmi les 72 disponibles. Les graphes suivants présentent les différents résultats pour nos modèles.



Ce premier graphique présente le score obtenu par tous les modèles

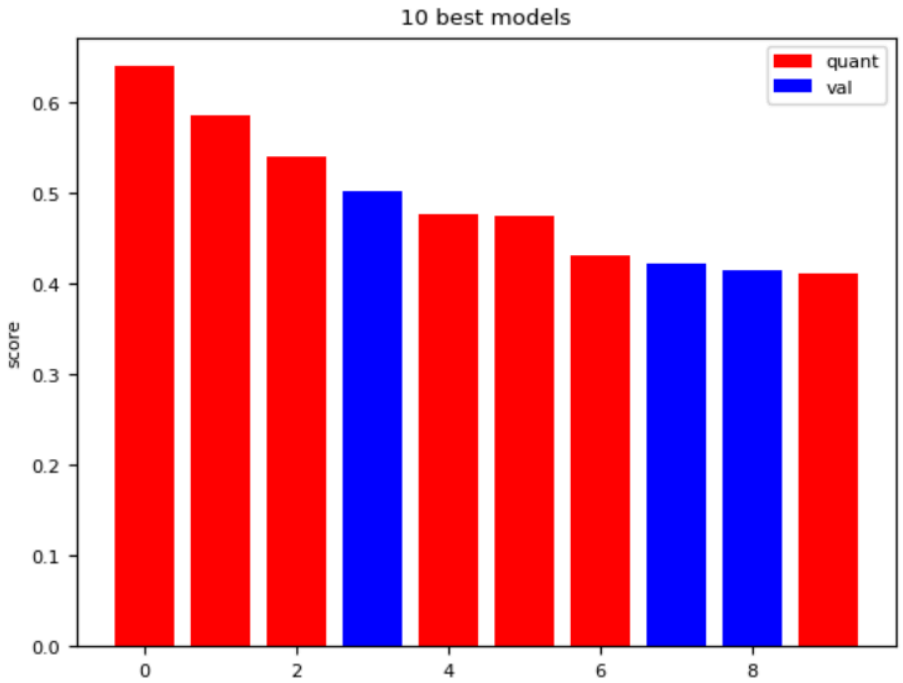


Ce second graphique présente le score en fonction de la taille de la fenêtre utilisée par le détecteur.



Ce dernier graphe présente la différence de score entre quantile et valeur pour les différents modèles

VI - Résultats et améliorations



Ce graphique permet de montrer les 10 meilleurs modèles ainsi que leur type.